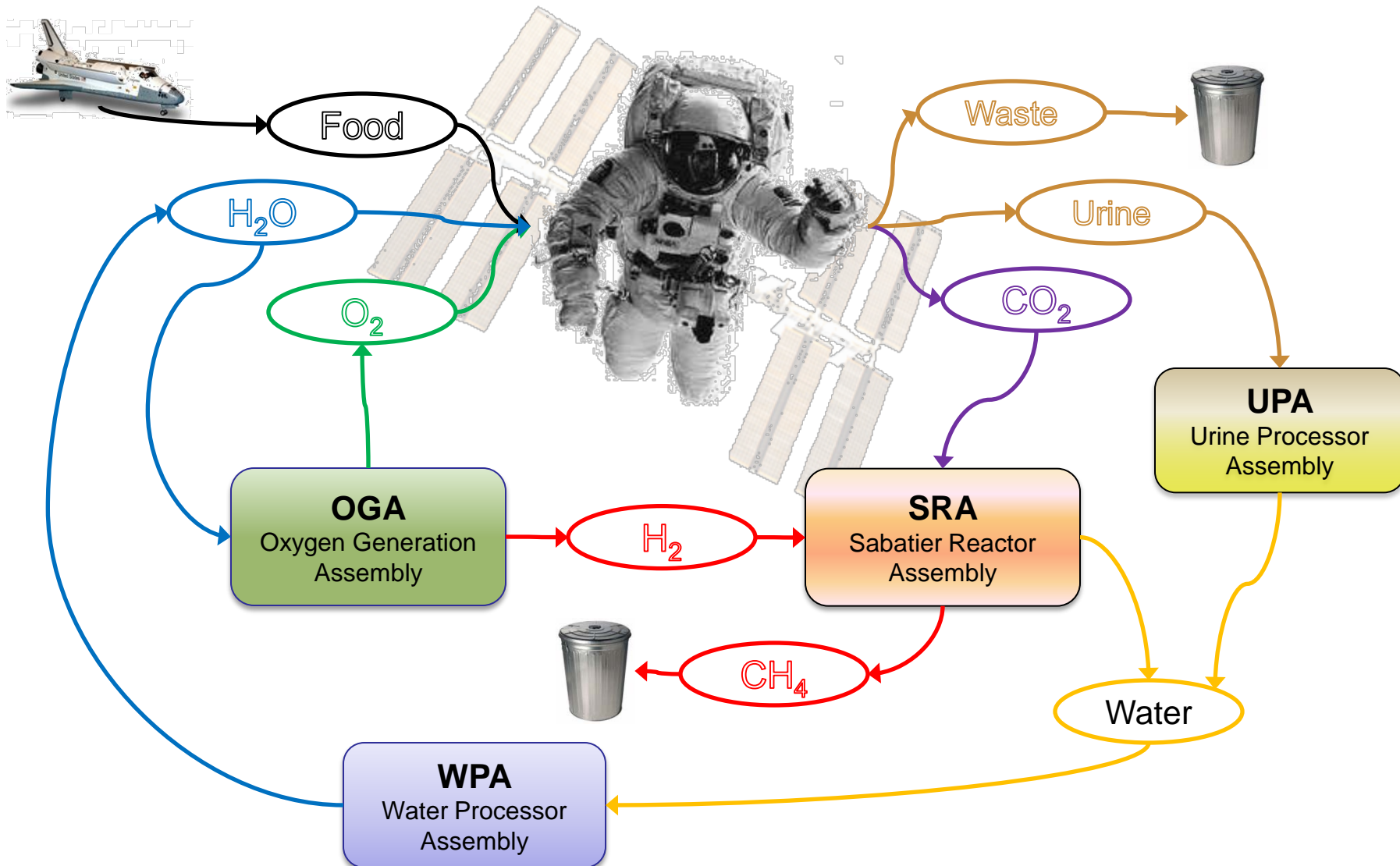
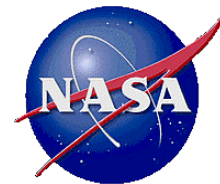


Executable Behavior Model of International Space Station (ISS) Urine Processor Assembly (UPA) Using MATLAB Simulink

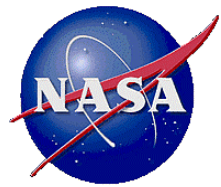
This technical data is controlled under the Export Administration Regulations (EAR) ECCN EAR 99, and may not be exported to foreign persons from Cuba, Iran, North Korea, Sudan and Syria, either in the U.S. or abroad, without a license or exception from the U.S. Department of Commerce.

Ryan Starn
9/17/2010

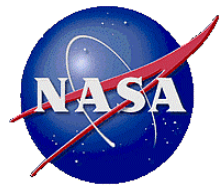
Regenerative Life Support Systems Overview



Reasons for Modeling UPA



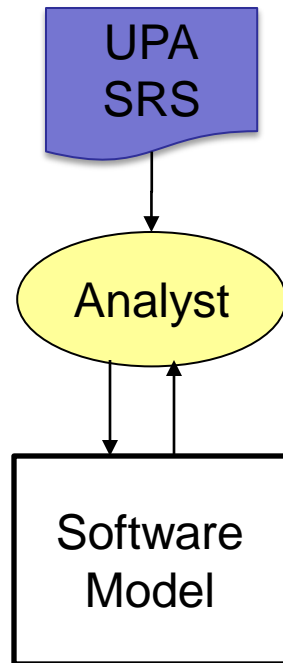
- IV&V Started Analysis Late in Life Cycle
 - UPA in Test Phase of Development
- IV&V had Deficient System Understanding
 - Design Documentation Missing
 - Test Documentation Incomplete
- UPA System Crucial for 6 Person Crew
 - Sustaining Engineering
- WPA & OGA Developed using MATRIXx
 - Design / Analysis / Testing
- Small System Size (~200 Software Requirements)



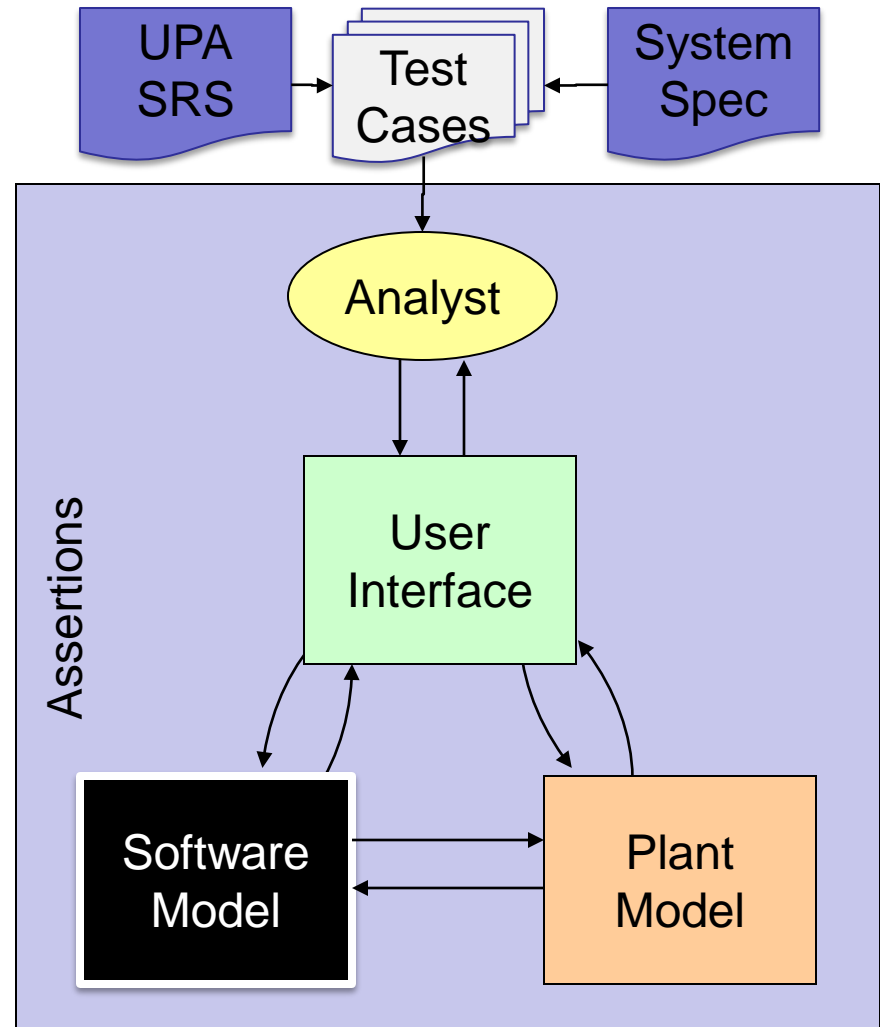
Modeling Objectives

- Gain System Understanding:
 - Uncover Requirement Deficiencies:
 - Completeness, Ambiguity
 - Uncover Design Impacts:
 - Data Consistency
 - Control Flow
 - Testability
 - Uncover Undesired Behavior:
 - Expand Developer formal testing.
 - Determine Conflicting & Unreachable States.
 - Determine Failure Scenarios.
 - Ensure system can handle multiple errors.
 - Verify long-term system operation.
- Static Analysis
- Dynamic Analysis

Model Analysis & Verification Techniques

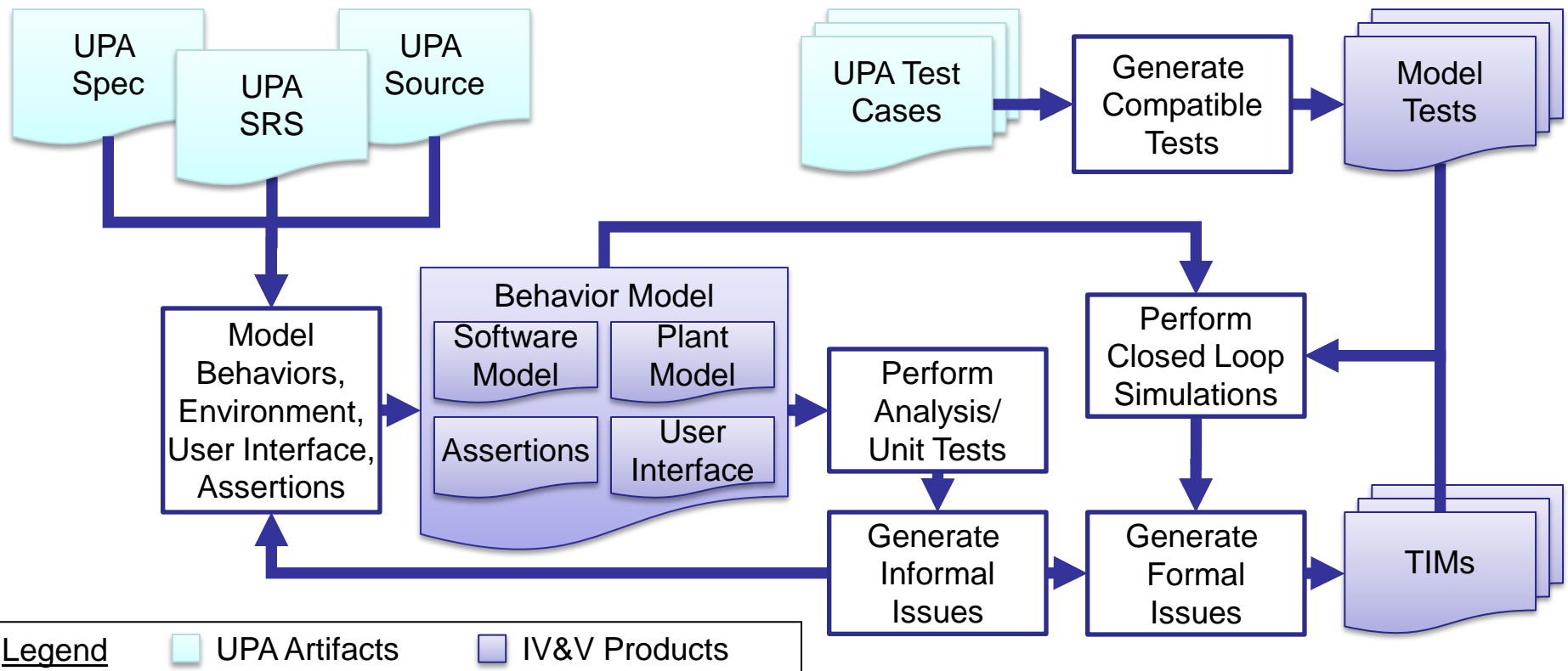


Static Analysis
& Unit Testing

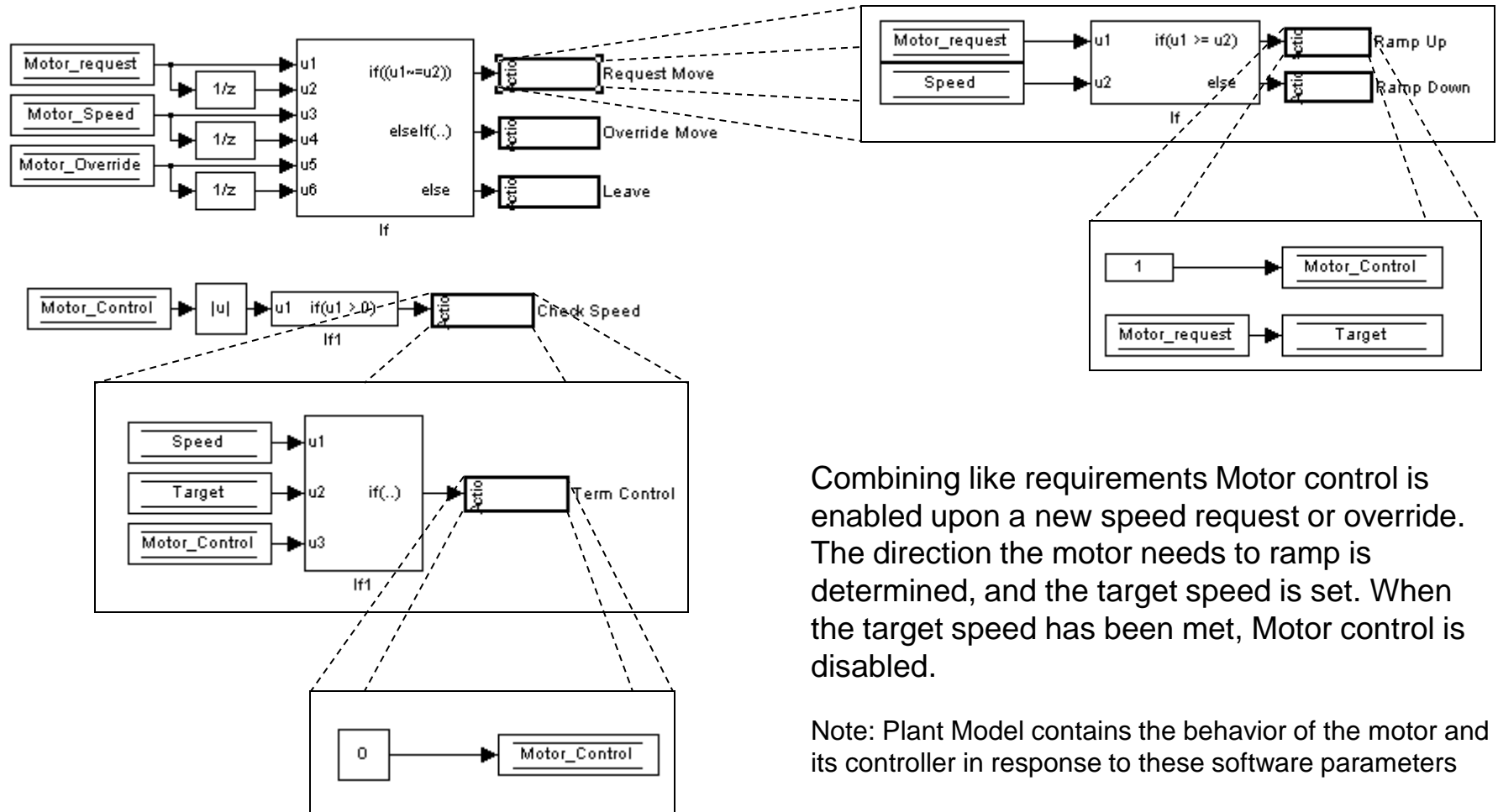


Dynamic Analysis: Closed Loop Testing

Modeling & Analysis Process



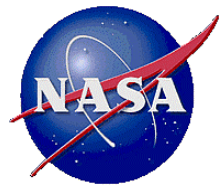
Software Model - Conditional Behaviors



Combining like requirements Motor control is enabled upon a new speed request or override. The direction the motor needs to ramp is determined, and the target speed is set. When the target speed has been met, Motor control is disabled.

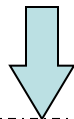
Note: Plant Model contains the behavior of the motor and its controller in response to these software parameters

Software Model - Mode Behaviors



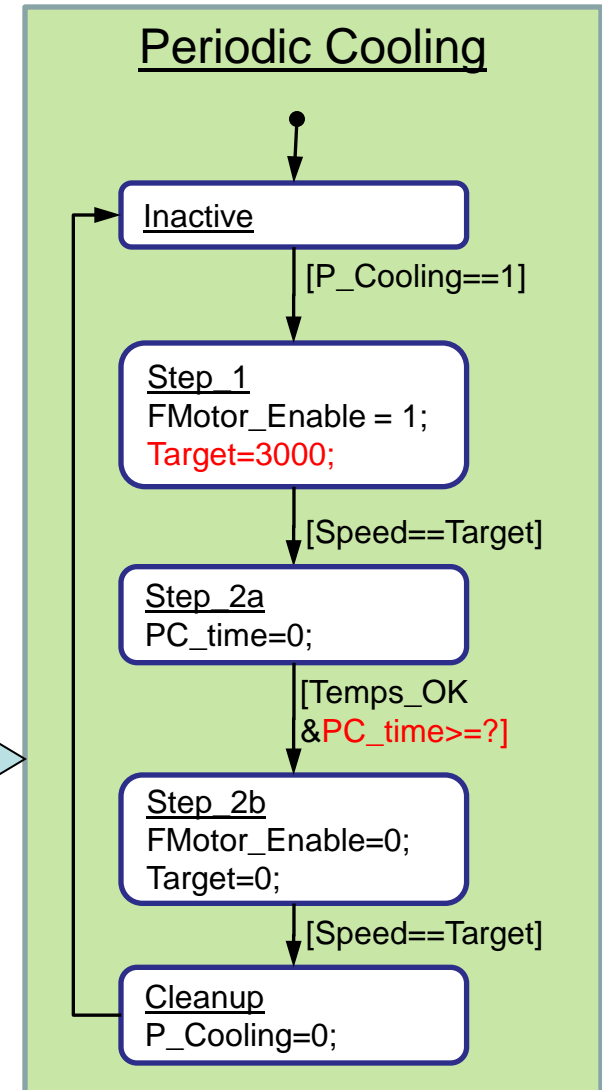
Periodic Cooling Cycle

- 1) The Software enables fan motor, ramps the motor up, and ensures that the target speed is achieved.
- 2) When the temperatures are below their setpoints and the fan motor has been operating for a minimum time, the software disables the motor, and ensures the motor stops.



Ambiguous Requirements

ramp rate - (behavior defined elsewhere)
 temperatures & setpoints – (behavior defined elsewhere)
target speed == 3000rpm. (only speed referenced in SRS)
minimum time == ?.



```

graph LR
    T1[Sensor_T] --> S[Sensor_Smoothing]
    S --> T2[Sensor_T]
    S --> O[sensor_out]

```

S-Function Builder: UPA_Model_2_2/UPA Software/Sensor Handling/Read Sensors/S-Function Builder16

Parameters

S-function name: Build

S-function parameters

Name	Data type	Value
------	-----------	-------

Code description

Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced as `xD[0]...xD[n]`, `xC[0]...xC[n]` respectively. Input ports, output ports and parameters should be referenced using the symbols specified in the Data Properties. These references appear directly in the generated S-function.

Code for the mdlOutputs function

```

/*sensor[smooth_value,smooth_enable,n_avg,valid_low,valid_high,history(),raw(),raw_new
raw_new: raw sensor reading
raw(): raw sensor reading history
history(): smooth history reading (within limits)
valid_high: valid high limit for a raw reading
valid_low: valid low limit for a raw reading
n_avg: the sliding average number of readings to use in calculations
smooth_enable: determines whether the average value is used or simply the previous
smooth_value: the sensor reading used for all limits and logic
The last entry is modified at each cycle of the software to store the raw reading an

```

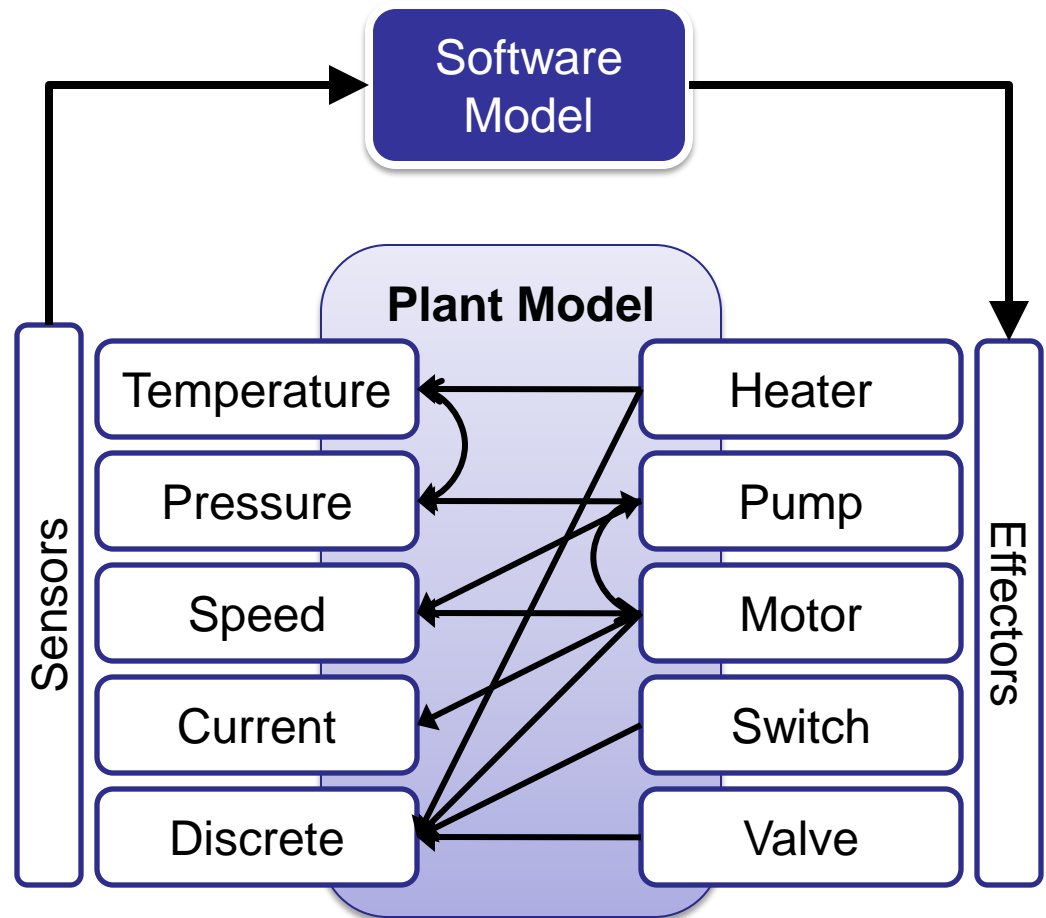
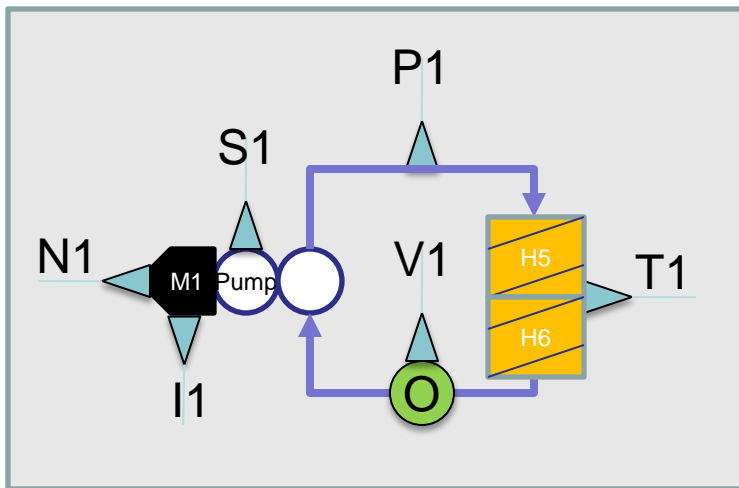
☒ Inputs are needed in the output function(direct feedthrough)

Cancel Help

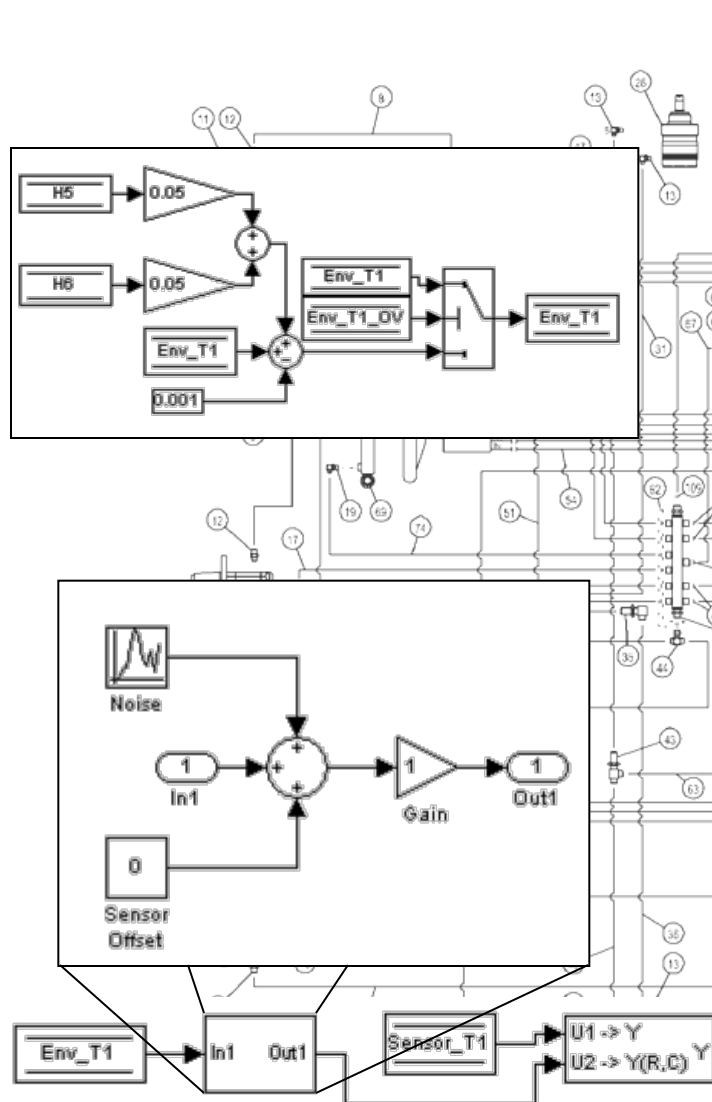
S-Functions contains custom code to process the sensor data and update the matrix

Plant Model

- Automates environment response to stimuli.
- Manipulates effectors in response to software commands.
- Generates ideal or noisy signals.
- Simulates hardware failures.



Plant Model - Behaviors

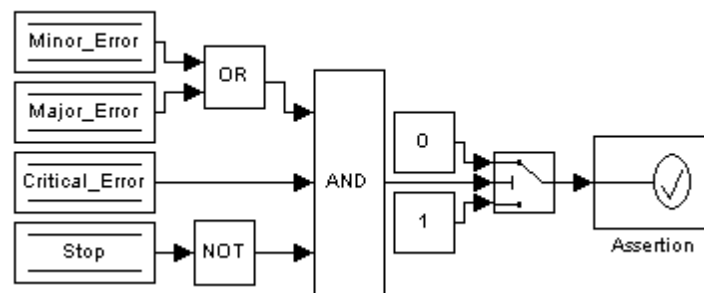


- Detailed Schematic, System Specifications and As-Run results were utilized to determine the nominal system response.
- Model behavior of the environment and system components in response to one another and Software commands.
 - Ex: When heaters H5 and H6 are on, temperature T1 will steadily increase, otherwise, T1 will slowly decrease.
- Retain capability to override Plant to simulate failure and special scenarios.
- Supply input to the Software Model via Sensors.
 - Signal can be ideal or realistic

Assertion Modeling

- Using the SRS and PIDS, develop models which represent each required aspect of the system or software.
- Run Assertions in parallel with the model.
- Model only requirements that cannot be verified directly using standard FQT test suites.

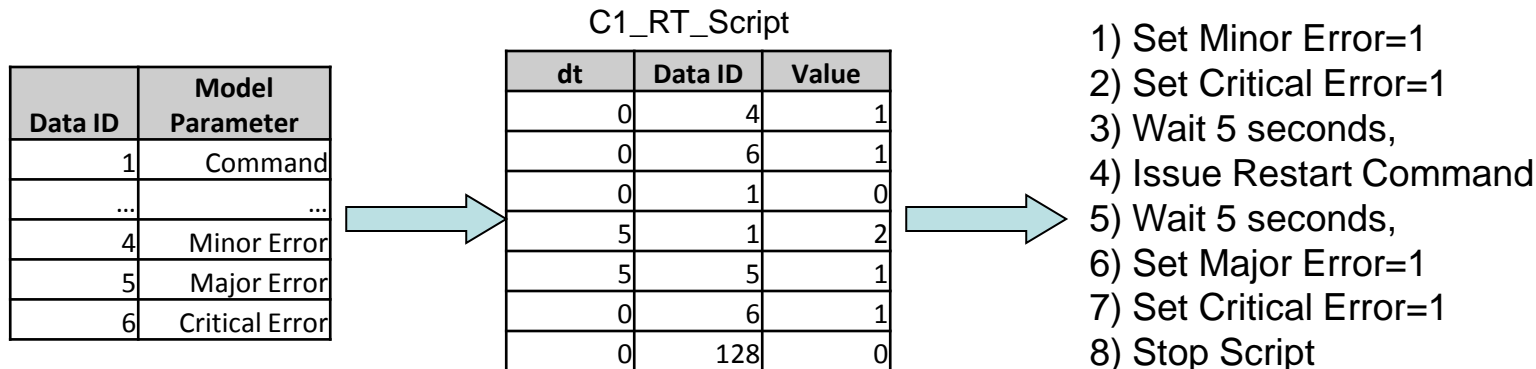
**A critical error will always cause the software to stop the system.
If a critical error is detected, it will be processed instead of other errors.**



Closed Loop Simulation Sample Test & Script

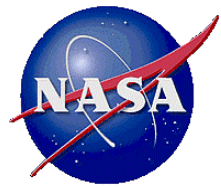
**A critical error will always cause the software to stop the system.
If a critical error is detected, it will be processed instead of other errors.**

Step #	Actions	Verifications	Pass/Fail Notes
1	Prepare Test_C1_RT_Script to run		N/A
2	Start Simulation	Verify System_State becomes "Startup"	Pass
3	Run Script	Verify System_State becomes "Stopped" and Critical Error is shown	State == Diagnostic Critical Set
4	Wait 5 seconds	Wait, verify System_State becomes "Ready"	Pass
5	Wait 5 seconds	verify System_State becomes "Stopped" and Critical Error is Shown	Pass
6	End simulation		



If any error occurs in Startup Mode, System will revert to Diagnostic Mode.

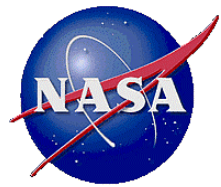
Requirement Issue – Conflicting Behavior



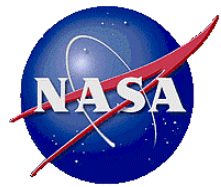
UPA Modeling Outcome

- High Severity Issue Uncovered (**Incomplete Requirement**).
 - Motor Remains Active After System Shutdown.
 - Motor is unmonitored and may result in crew injury or fire.
- 23 Lower Severity Issues Uncovered.
 - Test incomplete, incompatible, or results not expected.
- Traditional IV&V methods uncovered 1200 issues
 - 3 High, 942 Medium, and 255 Low Severity
- IV&V system knowledge utilized by program to complete software verification
 - 15 updated test procedures, 2 new test procedures, 41 inspections.

CDRA Behavior Model



- Carbon Dioxide Removal Assembly (CDRA)
 - extracts CO₂ from cabin atmosphere.
 - SRS is piecemeal collection of Change Notices.
 - No design or model available.
 - MATRIXx Autocode
- Modeling Objective:
 - Uncover SRS deficiencies and errors.
 - Uncover nominal/off-nominal failure scenarios
- Outcome:
 - Numerous SRS defects uncovered
 - Active BIT does not exercise software inhibit of effectors.
 - POST cannot complete due to missing requirement to honor ABIT commands while in Initialize Mode.



Lessons / Limitations

- Modeling improves IV&V system understanding
- Allows further analysis of off-nominal/failure scenarios
- Modeling does not replace traditional IV&V
- Model is hardware and language independent.
- Subject Matter Expert required for development/analysis
- Fidelity of model and cost need to be balanced.
- Modeling and Testing requires ~4hr/requirement
 - Applicability to larger systems is unproven